

12/15/99
JCS06 U.S. PTO

Please type a plus sign (+) inside this box → ☒

PTO/SB/05 (12/97)

Approved for use through 09/30/00. OMB 0651-0032

Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1.53(b))	Attorney Docket No.	4968-706	Total Pages	26
	First Named Inventor or Application Identifier			
	Jürgen Kienhöfer et al., "Method and Apparatus for Executing Multiple Java(TM) Applications on a Single Java(TM) Virtual Machine			
	Express Mail Label No.	EL341845433	US	

APPLICATION ELEMENTS See MPEP chapter 600 concerning utility patent application contents.		ADDRESS TO: Assistant Commissioner for Patents, Box Patent Application Washington, DC 20231	
1. <input type="checkbox"/> Fee Transmittal Form (Submit an original, and a duplicate for fee processing)		6. <input type="checkbox"/> Microfiche Computer Program (Appendix)	
2. <input checked="" type="checkbox"/> Specification [Total Pages 21] (preferred arrangement set forth below) <ul style="list-style-type: none"> - Descriptive title of the invention - Cross References to Related Applications - Statement Regarding Fed sponsored R&D - Reference to Microfiche Appendix - Background of the Invention - Brief Summary of the Invention - Brief Description of the Drawings - Detailed Description - Claim(s) - Abstract of the Disclosure 		7. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary) <ul style="list-style-type: none"> a. <input type="checkbox"/> Computer Readable Copy b. <input type="checkbox"/> Paper Copy (identical to computer copy) c. <input type="checkbox"/> Statement verifying identity of above copies 	
3. <input checked="" type="checkbox"/> Drawing(s) (37 CFR 1.152) [Total Sheets 1]		8. <input type="checkbox"/> Assignment Papers (cover sheet & documents(s))	
4. <input checked="" type="checkbox"/> Oath or Declaration (unsigned) [Total Pages 3] <ul style="list-style-type: none"> a. <input type="checkbox"/> Newly executed (original or copy) b. <input type="checkbox"/> Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with Box 17 completed) [Note Box 5 below] <ul style="list-style-type: none"> i. <input type="checkbox"/> DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b). 		9. <input type="checkbox"/> 37 CFR 3.73(b) Statement (when there is an assignee) <input type="checkbox"/> Power of Attorney	
5. <input type="checkbox"/> Incorporation By Reference (useable if Box 4b is checked) The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.		10. <input type="checkbox"/> English Translation Document (if applicable) 11. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 12. <input type="checkbox"/> Preliminary Amendment 13. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) (Should be specifically itemized) 14. <input type="checkbox"/> Small Entity Statement(s) <input type="checkbox"/> Status still proper and desired 15. <input type="checkbox"/> Certified Copy of Priority Document(s) (if foreign priority is claimed) 16. <input type="checkbox"/> Other:	

16. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No. _____

17. CORRESPONDING ADDRESS			
<input checked="" type="checkbox"/> Customer Number or Bar Code Label		021971	
<input type="checkbox"/> Correspondence address below		(Insert Customer No. or Attach bar code label here)	
NAME			
ADDRESS			
CITY	STATE	ZIP CODE	
COUNTRY	TELEPHONE	FAX	

SUBMITTED BY
 Typed or
 Printed Name Kent R. Richardson
 Signature

Reg. Number 39,443
 Date December 15, 1999

UNITED STATES PATENT APPLICATION

OF

DR. JÜRGEN KIENHÖFER AND RANJIT DESHPANDE

FOR

METHOD AND APPARATUS FOR EXECUTING MULTIPLE JAVA(TM)
APPLICATIONS ON A SINGLE JAVA(TM) VIRTUAL MACHINE

PREPARED BY WILSON SONSINI GOODRICH & ROSATI

RELATED APPLICATIONS

This application relates to, incorporates by reference, and claims priority from, United States Provisional Patent Application Serial Number 60/112,803 entitled, "Method and Apparatus for Executing Multiple JAVA Applications on a Single JAVA Virtual Machine", filed December 18, 1998, having inventor Dr. Jürgen G. Kienhöfer.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of improved execution environments for software applications running in the JAVA(TM) language. In particular, the invention relates to improvements designed to support the operation of multiple unmodified JAVA(TM) applications on an unmodified JAVA(TM) Virtual Machine.

Description of the Related Art

1. Description of Problem

JAVA(TM) applications are transportable byte codes that can be executed on a number of platforms. The execution environment for JAVA(TM) applications is a JAVA(TM) Virtual Machine (JVM). For each platform a JVM must be available to execute the JAVA(TM) applications. The specification of, and the implementation of, a JVM is described in documents such as "JAVA(TM) Virtual Machine", John Meyer and Troy Downing, O'Reilly and Associates, 1997. Additionally, the book "The JAVA(TM) Virtual Machine

Specification”, Tim Lindholm and Frank Yellin, Addison Wesley, 1997, also describes a JVM.

For each JAVA(TM) application that a user wishes to run on a JAVA(TM) Virtual Machine, a separate JVM must be run together with a
5 separate execution environment. This execution environment includes an object store in memory, also referred to as a JAVA(TM) heap, for storing JAVA(TM) objects and data. Also, the environment includes a “garbage collector” that deletes unused objects and compacts the JAVA(TM) heap. This arrangement consumes large amounts of system memory on each JVM, and thus each
10 application. When used, as JAVA(TM) was originally designed on a client computer, this is not a problem as a user is typically running only one or two applications at a time. Further, the user’s computer is dedicated to running those applications.

In contrast, server computers may require that multiple JAVA(TM)
15 applications be running simultaneously. For example, a server might be deployed to handle processing for a large number of client computers. If the JAVA(TM) standard is followed, each of running JAVA(TM) application on the server would need to run in separate JVM’s, each with an associated execution environment. Thus, each application would have its own JAVA(TM)
20 heap and separate garbage collection process. The multiplicity of the garbage collection processes across all of the JVM’s can consume significant amounts of processor time and lead to decreased performance.

The JAVA(TM) Virtual Machine could be rewritten in its entirety to support multiple applications at one time. However, such an approach would

require major re-architecting of the JAVA(TM) and/or JVM standards and specifications. Additionally, JAVA(TM) applications could be rewritten in source code to support multiple applications on a single JVM.

2. Prior Art JAVA(TM) Execution Environment

5 Turning to Figure 1, and the prior art JAVA(TM) execution environment, the execution environment includes a JAVA(TM) Virtual Machine 100, including JAVA(TM) base classes 102 and a primordial class loader 104. An optional application class loader 106 is depicted as well as a single JAVA(TM) application 108. This is the typical JAVA(TM) execution
10 environment according to the prior art.

In the normal operation of a JVM, a JAVA(TM) application compiled to run on the JVM arrives in a sequence of byte codes arranged in class files. The class files, which can be remotely or locally accessed, are loaded by class loaders and executed in a threaded environment by the JVM. Execution of
15 JAVA(TM) applications take place in threads, which are part of thread groups, and invokes calls to methods of the associated objects. Each thread that is created from a thread in a given thread group also belongs to that thread group. Executions of thread causes the creation of objects, which are stored in portions of the JAVA(TM) heap during run time. An application is able to run on a
20 JAVA(TM) execution environment with a large set of JAVA(TM) base classes, which are sometimes considered part of the JVM. The base classes received calls from the application to enable many basic functions.

Object creation during execution within the JVM utilizes a class loader architecture. There are two types of class loaders in the JAVA(TM) execution

environment. The first type is a “primordial” class loader, e.g. the primordial class loader 104. The primordial class loader is considered part of the JVM itself and is designed to load certain class loaders. Usually the primordial class loader 104 is used to load classes of the application.

5 Another type of class loader is available for loading objects. This type of class loader is a JAVA(TM) class loader object written in JAVA(TM). This type of class loader can be installed by a JAVA(TM) application into a thread. When this type of class loader is installed into a thread other application objects within that thread are loaded using that class loader.

10 Notably the prior art JVM does not allow for a hierarchy of application class loaders. Thus, a JAVA(TM) application such as the JAVA(TM) application 108 cannot install additional application class loaders.

3. Conclusion

15 The prior techniques do not permit the execution of multiple unmodified JAVA(TM) applications on a single unmodified JAVA(TM) Virtual Machine. Further, the prior techniques do not support shared usage of the JAVA(TM) base classes by applications running on the JVM. Accordingly, what is needed is a method and apparatus for supporting multiple unmodified JAVA(TM) applications on a single JVM using a single copy of the base classes for all of
20 the JAVA(TM) applications.

SUMMARY OF THE INVENTION

A modified JAVA(TM) execution environment is described. The modified environment supports multiple JAVA(TM) applications on a single

JAVA(TM) virtual machine (JVM). This modified environment provides significant memory and performance improvements when running multiple applications on a single computer system. Notably, no changes are needed to the source code of an application to take advantage of the modified environment.

- 5 Further, embodiments of the invention may support shared access to base classes through the use of overlays. Additionally, system resource permissions can be enforced based upon the user permissions associated with a running application. Notably, embodiments of the invention allow multiple applications to share the abstract window toolkit (AWT) on a per display basis. Since only a
- 10 single garbage collection routine is necessary, applications see improved performance relative to running in different JVMs. Further, the shared base classes eliminate significant memory overhead.

- According to some embodiments of the invention, a class loader and a thread group is dynamically generated for each application to be run in the
- 15 modified environment. This class loader defines a name space for the application. Additionally, the thread group defines the set of threads for that application. The class loader also loads the application classes into the JVM. The application itself can have an application class loader, an application security manager, and/or create additional thread groups.

- 20 As necessary, when the overlaid base classes are called, the calling application can be determined. Two approaches are used by some embodiments of the invention. In the first approach, the class loader of the calling method is determined. This in turn allows the identification of the application through reference to data associated with the class loader. Another approach is to scan

the thread group hierarchy of the JVM to identify a thread group with which application information has been associated by the class loader. Once the application is identified, the underlying base class functionality can be implemented using the appropriately selected files, resources, variable values,

5 etc.

Additionally, embodiments of the invention can support system resource permissions, e.g. user access rights, on a per application basis. Each application can be associated with a user. The overlaid base

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2

BRIEF DESCRIPTION OF THE FIGURES

Fig. 1 illustrates a JAVA(TM) Virtual Machine environment according to the prior art.

Fig. 2 illustrates a JAVA(TM) Virtual Machine environment according to one embodiment of the invention.

Fig. 1 illustrates a JAVA(TM) Virtual Machine environment according to the prior art.

DETAILED DESCRIPTION

The modified JAVA(TM) execution environment supported by
embodiments of the invention will be described with reference to Figures 1 and
2. Figure 1 shows the JAVA(TM) execution environment according to the prior
art and was described above. Figure 2 shows the JAVA(TM) execution
environment according to one embodiment of the invention and will now be
described in greater detail.

Figure 2 shows the modified JAVA(TM) execution environment
according to one embodiment of the invention. Elements of figure 2 that are
found in figure 1 are designated with the same reference numerals. For example,
Figure 2 includes the JVM 100. The JVM 100 used in Figure 2 may be identical
to the JVM 100 used in Figure 1, but should at least be a substantially
unmodified JVM.

The term "substantially unmodified" as used in this application refers to
a JVM or JAVA(TM) application suitable for use in the prior art JAVA(TM)
execution environment of Figure 1. For example, a JVM supporting just in time
(JIT) that can execute substantially unmodified JAVA(TM) applications would
be a substantially unmodified JVM. One further example may be instructive. A
JAVA(TM) application 108 is substantially unmodified, if it can be used in the
execution environment of Figure 1 without the need for source code – or byte
code – modifications to run in the execution environment of Figure 2.

Examples of substantially unmodified JVMs usable according to
embodiments of the invention include JVMs from Sun Microsystems, Mountain
View, California; JVMs from Microsoft Corporation, Redmond, Washington;

JVMs from Apple Computer Corporation, Cupertino, California; and/or other available JVMs. For the purposes of this discussion it will be assumed that the JAVA(TM) Virtual Machine complies with a JAVA(TM) standard and that the JAVA(TM) applications similarly comply with a JAVA(TM) standard.

5 The elements of Figure 2 will now be described in greater detail. The substantially unmodified JVM 100 supports the modified execution environment of Figure 2. The substantially unmodified JVM 100 includes base classes 102. The base classes 102 are substantially unmodified base classes suitable for use in a standard JAVA(TM) execution environment such as the
10 JAVA(TM) execution environment of Figure 1.

 Additionally, Figure 2 includes base class overlays 200. The base class overlays 200 provides support for multiple JAVA(TM) applications using only a single copy of the base class 102. The base class overlays 200, allow multiple applications to reference the base classes 102 without conflicts due to different
15 access privileges and/or base class definitions that inhibit sharing. The base class overlays 200 will be described in further detail below. The modified JAVA(TM) execution environment also includes a primordial class loader 104 that is substantially unmodified and suitable for use according to the prior art JAVA(TM) execution environment.

20 The modified JAVA(TM) execution environment includes a multiple application class loader 206. This class loader provides support for multiple applications. Additionally, a security manager 204 provides for different degrees of access to different applications based on privileges. The multiple

application class loader 206 handles the class loading of JAVA(TM)
applications within the modified execution environment of Figure 2.

Compare this to the standard execution environment of Figure 1, where
the primordial class loader 104 would load the JAVA(TM) application 108.

- 5 According to the modified execution environment of Figure 2, the multiple
application class loader 206 would invoke the JAVA(TM) application 108.

In order to support the launch of multiple applications, a launch interface
202 is provided. The launch interface 202 may itself be a JAVA(TM)
application. The launch interface 202 may provide a command line interface or
10 other interface for invoking the execution of JAVA(TM) applications within the
modified execution environment of Figure 2. The launch interface 202 may
itself be loaded by the multiple applications class loader 206 as a JAVA(TM)
application running within the modified JAVA execution environment. In some
embodiments, the launch interface may respond to remote procedure
15 invocations, or some other type of message, and execute applications according
to parameters specified in the message. In some embodiments, the launch
interface 202 provides a UNIX-style command line interface with log in and
security procedures.

- The depiction of the multiple application class loader 206 as a single
20 class loader for all applications is a simplification. In fact, a class loader is
dynamically generated for each application. Thus, the launch interface 102, the
JAVA(TM) application 108, and the JAVA(TM) application 108B each has a
dynamically generated multiple application class loader 206 responsible for
loading the appropriate application classes. Each of the dynamically generated

multiple application class loaders can define its own namespace within which the loaded applications will execute.

As shown in Figure 2, once invoked from the launch interface 202, JAVA(TM) applications (e.g. the JAVA(TM) application 108) can have their
5 respective classes loaded within the modified execution environment of Figure 2. Similarly a second application, the JAVA(TM) application 108B could be loaded within the modified execution environment of Figure 2. These two applications would be sharing the same JAVA(TM) Virtual Machine 100 and the same base classes 102. However, the multiple application class loader 104
10 would place them in separate namespaces and would place them in different thread groups. The base class overlays 200 ensure appropriate behavior of the base classes 102 for each of the applications.

Notably, neither the JAVA(TM) application 108 nor the JAVA(TM) application 108B need to be modified at the source code level to operate within
15 the modified execution environment. The environment of Figure 2 is transparent to JAVA(TM) applications running in the environment.

The modified execution environment of Figure 2 only needs a single garbage collection process and a single copy of the base classes 102. This provides significant memory and speed savings. In some experiments, this
20 reduced the memory overhead to enable execution of over one hundred JAVA(TM) applications on a single JVM – on a single server computer, which would otherwise support only fourteen of these applications at the same time.
See also, Jürgen G. Kienhöfer, “Java Junction: Perkup, SCO Server Side Java

Technology", in SCO Coredump, Summer 1999, Number 13, page 8, also available at <<http://www.sco.com/developer/core13/perkup.htm>>.

The security manager 204 is an addition to the inherent security models of JAVA(TM). Prior art JVMs were typically invoked on client machines by a specific user and the single application ran with that user's privileges. In contrast, the execution environment of Figure 2 would typically be invoked with system privileges such as "root" on UNIX-like systems. As a result, each running application would, without additional security, be capable of accessing the entire system. Therefore, a security manager 204 can be provided to enforce operating system security – or other security – requirements.

In one embodiment of the invention, the security manager 204 uses parameters provided via the launch interface 202 to control the permissions granted to running applications. For example, if using the launch interface 202 an application is invoked using the privileges of "user 1", the security manager 204 would enforce operating system file permissions and resource permissions for that application according to the privileges granted to "user 1". Examples of enforced requirements include those for:

- reading, writing, creating, deleting, modifying, or examining system resources such as files and sockets;
- listening or accepting network connections to a reserved port;
- executing a program on the system or starting a sub process
- terminating the JAVA(TM) run time environment of Figure 2
- loading dynamic libraries and native methods.

Thus if the permissions on a particular file “x” indicate that it is owned by “user 2” and not readable by other users, an attempt by a JAVA(TM) application running as “user 1” to read the file may be denied.

- Each application running in the environment of Figure 2 may also have its own security management policies — for example, a set of JAVA(TM) sandbox policies.

Part of the base class overlays 200 involves the separation of certain resources that are not effectively shared between different programs. For example, the standard java.lang.system class uses static variables to define the input, output and errors streams. As a result, it is not possible to share that class without modification of the base classes. In this instance, the base class can be overlaid with modifications that can use two possible processes – possibly in conjunction with one another - to determine the current application and provide appropriate access to the shared base class.

- One process used by overlaid, or shared, classes to identify the correct application is to identify the class loader for the calling thread. If the class loader is an instance of the dynamically generated multiple application class loader, then it together with the namespace can be used to identify the application. Consequently, the correct resource permissions, list of accessible files, input and output devices, etc., are identified for use by the shared class.

The above approach may fail if the class loader for the object accessing the overlaid class is the primordial class loader 104. In that instance, the associated namespace may not provide adequate information to suitably identify the application and needed information.

Therefore, a second approach to determining the calling process can be used. In this case, the thread data structures within the JVM 100 can be examined to determine the calling object's thread. Then, the group for the thread can be identified. Information associated with the thread group about the application and its properties can then be identified. If necessary, the thread group hierarchy can be recursively examined until a thread group is found that is associated with information about the process.

As seen in the execution environment of Figure 2, the multiple application class loader 206 does extend the JVM 100 to support application class loaders in addition to the multiple application class loader 206. In some embodiments, it is necessary to configure the JVM not check for multiple class loaders to enable this capability. In other embodiments this change is not necessary if the JVM itself already supports hierarchies of class loaders.

The base class overlays 200 may involve adding checks for resource permissions. For example, the procedures for reading file must be overlaid to include identification of the user for the application, as described above, as well as verification of the user's rights with respect to that file. These changes to the base classes may be implemented in the base class overlays 200, in the security manager 204, or in a combination of the two.

The terms "program", or "computer program", as used in this application, refers to any sequence of instructions designed for execution on a computer system. A program may include a subroutine, a function, a procedure, an object method, an object implementation, an executable application, an

applet, a servlet, a source code, an object code, and/or some other sequence of instructions designed for execution on a computer system.

The base class overlays 200, the multiple application class loader 206, and the security manager 204 may be embodied as one or programs included in
5 one or more computer usable media such as CD-ROMs, floppy disks, or other media.

Some embodiments of the invention are included in an electromagnetic wave form. The electromagnetic waveform comprises information such as base class overlays, a multiple application class loader, and a security manger for use
10 in a modified JAVA(TM) execution environment. The electromagnetic waveform may include the multiple application class loader accessed over a network.

The foregoing description of various embodiments of the invention has been presented for purposes of illustration and description. It is not intended to
15 limit the invention to the precise forms disclosed. Many modifications and equivalent arrangements will be apparent.

CLAIMS

What is claimed is:

- 1 1. A modified JAVA(TM) execution environment comprising:
2 a substantially unmodified JAVA(TM) Virtual Machine,
3 a set of substantially unmodified base classes;
4 one or more overlays to the set of substantially unmodified base classes,
5 the one or more overlays enabling corresponding base classes to
6 support shared access by one or more substantially unmodified
7 JAVA(TM) applications;
8 an unmodified primordial class loader for loading the system base
9 classes as overlaid by the one or more overlays to the base classes;
10 a security manager supporting multiple applications and for limiting
11 access to system resources according to user permissions; and
12 an dynamic class loader generator for creating a class loader for loading
13 an application, the application classes and creating a thread group for
14 the application.
- 1 2. The modified JAVA(TM) execution environment of claim 1, wherein
2 the application includes at least one of an application class loader and an
3 application security manager.

1 3. The modified JAVA(TM) execution environment of claim 1, wherein
2 the one or more overlays include overlays to file classes to limit access to
3 system resources according to user permissions associated with the application.

1 4. The modified JAVA(TM) execution environment of claim 1, wherein
2 the application includes at least one of an application class loader and an
3 application security manager.

1 5. The modified JAVA(TM) execution environment of claim 1, wherein
2 the application includes one or more invocations of Abstract Window Toolkit
3 (AWT) classes.

1 6. The modified JAVA(TM) execution environment of claim 1, wherein
2 the one or more overlays support determining a calling application.

1 7. The modified JAVA(TM) execution environment of claim 6, wherein
2 the determining a calling application comprises identifying the class loader of a
3 calling method, and using the class loader to identify the application.

1 8. The modified JAVA(TM) execution environment of claim 6, wherein
2 the determining a calling application comprises identifying the thread group for
3 a calling method, and using the thread group to identify the application.

1 9. A method of supporting a number of applications in a single JAVA(TM)
2 execution environment, the method comprising:

3 means for generating a class loader for each of the applications in the
4 number of applications, the class loader providing a name space for
5 each application, and a thread group for each application;
6 means for overlaying one or more substantially unmodified base classes
7 to support the number of applications; and
8 means for determining a calling application for a method.

1 10. The method of claim 9, wherein at least one of the number of
2 applications includes an application class loader.

1 11. The method of claim 9, wherein at least one of the number of
2 applications includes an application security manager.

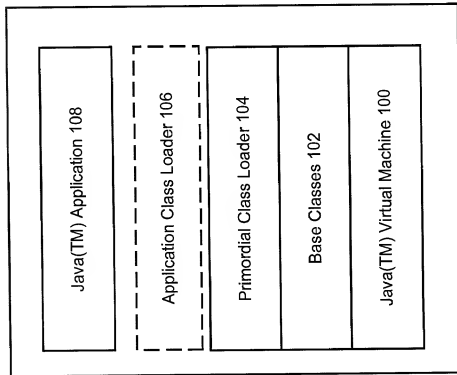
1 12. A computer data signal embodied in a carrier wave comprising:
2 a computer program for supporting a number of substantially
3 unmodified JAVA(TM) applications on a substantially unmodified
4 JAVA(TM) Virtual Machine (JVM), the JVM including a set of
5 substantially unmodified base classes and a substantially unmodified
6 primordial class loader, the program comprising:
7 a first set of instructions for generating a class loader for each of the
8 JAVA(TM) applications in the number of JAVA(TM)
9 applications, the class loader providing a name space for each
10 application, and a thread group for each application;
11 a second set of instructions for overlaying one or more substantially
12 unmodified base classes to support the number of applications;
13 and

14 a third set of instructions for determining a calling application for a
15 method.

1 13. The computer data signal of claim 12, wherein the first set of
2 instructions further associates a user with each application, and wherein the
3 program further comprises a fourth set of instructions for limiting access to a
4 system resource by an application according to whether the user associated with
5 the application has access to the system resource.
6

ABSTRACT

A modified JAVA(TM) execution environment is described. The modified environment supports multiple JAVA(TM) applications on a single JAVA(TM) virtual machine (JVM). This modified environment provides significant memory and performance improvements when running multiple applications on a single computer system. Notably, no changes are needed to the source code of an application to take advantage of the modified environment. Further, embodiments of the invention may support shared access to base classes through the use of overlays. Additionally, system resource permissions can be enforced based upon the user permissions associated with a running application. Notably, embodiments of the invention allow multiple applications to share the abstract window toolkit (AWT) on a per display basis. Since only a single garbage collection routine is necessary, applications see improved performance relative to running in different JVMs. Further, the shared base classes eliminate significant memory overhead.



(Prior Art) **Fig 1**

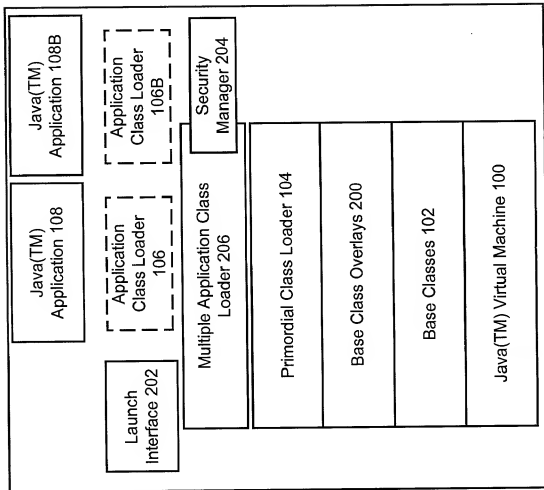


Fig 2

**COMBINED DECLARATION AND POWER OF ATTORNEY
FOR UTILITY PATENT APPLICATION**

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

**METHOD AND APPARATUS FOR EXECUTING MULTIPLE JAVA(TM)
APPLICATIONS ON A SINGLE JAVA(TM) VIRTUAL MACHINE**

the specification of which

 X is attached hereto.

 was filed on as Application No.
and was amended on *
(If Applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a) which states in relevant part: "Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section....The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98."

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate as indicated below and have also identified below any foreign application for patent or inventor's certificate on this invention having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

Priority Claimed

(Number)

(Country)

(Day/Month/Year Filed)

Yes No

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s), and under §119(e) of any United States provisional application(s), listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulation, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

60/112,803
(Application Serial No.)

December 18, 1998
(Filing Date)

Pending
(Patented, Pending, Abandoned)

(Application Serial No.)

(Filing Date)

(Patented, Pending, Abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith, and to file, prosecute and to transact all business in connection with international applications directed to said invention:

Paul Davis	29,294
John J. Bruckner	35,816
David J. Weitz	38,362
Kent R. Richardson	39,443
David J. Abraham	39,554
U.P. Peter Eng	39,666
George A. Willman	41,378
Jinntung Su	42,174
Van Mahamedi	42,828
Shantanu Basu	43,318
Barbara B. Courtney	42,442
Richard L. Gregory, Jr.	42,607

Address all correspondence to:

Kent R. Richardson
Wilson Sonsini Goodrich & Rosati
650 Page Mill Road
Palo Alto, CA 94304

Direct all telephone calls to Erik Oliver at (650) 493-9300.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these

statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, §1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or
first inventor:

Jürgen Kienhöfer

Inventor's signature:

Date:

Citizenship:

German

Residence:

140 Walti Street, Santa Cruz, California 95060

Post Office Address:

Same as above

Full name of second joint
inventor, if any::

Ranjit Deshpande

Inventor's signature:

Date:

Citizenship:

Indian

Residence:

755 14th Ave., Apt. 401, Santa Cruz, California 95062

Post Office Address:

Same as above